On the Combination of Coevolution and Novelty Search

On the Combination of Coevolution and Novelty Search

Fabian Fränz*, Jan Paredis†, Rico Möckel‡
Department of Data Science and Knowledge Engineering, Maastricht University,
P.O. Box 616, 6200 MD Maastricht, The Netherlands
Email: *f.fraenz@t-online.de, †j.paredis@maastrichtuniversity.nl, ‡rico.mockel@maastrichtuniversity.nl

***Abstract*—This paper develops a new method for coevolution, named Fitness-Diversity Driven Coevolution (FDDC). This approach builds on existing methods by a combination of a (predator-prey) Coevolutionary Genetic Algorithm (CGA) and novelty search. The innovation lies in replacing the absolute novelty measure with a relative one, called Fitness-Diversity. FDDC overcomes problems common in both CGAs (premature convergence and unbalanced coevolution) and in novelty search (construction of an archive). As a proof of principle, Spring Loaded Inverted Pendulums (SLIPs) are coevolved with 2D-terrains the SLIPs must learn to traverse.**

I. Introduction

The seminal work of Hillis [1] showed how predator-prey *coevolution* could be used for optimisation. Inspired by this work, numerous authors have applied coevolution across multiple domains. At the same time, it has become clear that coevolution has a number of potential shortcomings. One problem is that coevolution may not always reach a global optimum, e.g., *cycling* may occur. Another is that it may suffer from *premature convergence*, which typically results in a loss of diversity [2]. Third, the coevolution can become *unbalanced* if evolving populations diverge from one another [3] and one population becomes too strong too quickly, preventing a gradual stepwise arms race between the populations. This fitness divergence results in a lack of gradient and thus a halt to evolution. To ameliorate these problems, several diversity preserving methods have been proposed, including fitness sharing [4] and crowding [5], [6]. Alternatively, Rosin and Belew [7] proposed the use of an archive of old individuals, called *Hall of Fame*.

Another approach is suggested by Lehman and Stanley [8], who introduce *novelty search*, a different evolutionary force which does not directly involve fitness but instead uses novelty, a behavioural distance measure between current and past individuals. This successfully allows a novelty-driven genetic algorithm (GA) to evolve solutions that do not follow the actual objective function and therefore, cannot be deceived by it. However, their algorithm suffers from the drawback that a history of past individuals must be maintained. This paper builds upon their findings while avoiding the need to maintain such a history.

This paper integrates novelty search in the Coevolutionary Genetic Algorithm (CGA) introduced by Paredis [9]. The structure of the CGA allows for the integration of novelty search in an easy and straight-forward way. To illustrate this, a simple coevolutionary application is used to demonstrate the implications of this integration and to provide a proof of principle of the complementarity of coevolution and novelty search.

The application used here, is the control of *spring loaded inverted pendulums* (SLIPs) over 2D-terrains [10]. A SLIP consists of a stiff spring attached to a point mass and is often used to model walking, running and jumping in monoped [11], biped [3] and multiped [12] simulations. A SLIP usually has a single control parameter (its angle of impact) and three dependent morphological parameters (mass, length and stiffness). Thus the SLIP follows the motion cycle as shown in figure 1. It has been shown [11] that SLIPs already exhibit self-stabilisation without active control; however, it is rather difficult to control them on rough terrain. In this paper, SLIP controllers are coevolved with terrains the SLIPs must traverse.

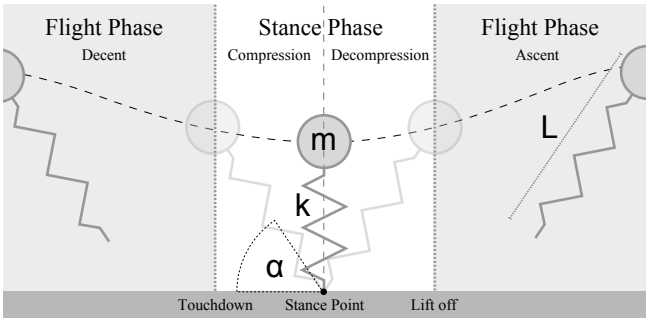

Fig. 1. A SLIP in motion. m = mass, L = rest length, k = spring constant, α = angle of attack.

The SLIP population uses a standard fitness evaluation, while the terrains use novelty search. The proposed algorithm, which is called Fitness-Diversity Driven Coevolution (FDDC), is then compared with the standard CGA where both populations are fitness based.

The paper is structured as follows: First, the general design of a CGA is described. Next, the idea behind novelty search is elaborated. Section IV describes FDDC and explains how adding novelty to CGAs achieves Fitness-Diversity. The following section provides experimental results obtained via FDDC. Section VI discusses the results and a section of future research is provided. Finally, conclusions are drawn.

978-1-5090-4601-0/17/$31.00

II. Novelty Search

In their initial work on novelty search, Lehman and Stanley [8] showed how an objective function can be misleading and why it can be best to ignore the objective altogether. In order to reduce the algorithmic complexity for solving difficult optimisation problems, simple objective functions approximate the distance to the optimisation goal. This simplicity, however, comes at a cost.

One of the problems of optimisation is that most heuristics include *dead-ends* in their solution-space. Consider navigating a maze: a simple objective function could be to minimise the Euclidean distance to the maze exit. Here, the distance could be iteratively reduced by moving in the direction to the goal. Unfortunately, there often exists paths through a maze that initially point to the goal but later become dead-ends. Most optimisation algorithms find such dead-ends difficult to escape – they have been deceived by a poorly specified objective function.

Novelty search avoids deception by ignoring the objective altogether. It does this by developing solution attempts, which are simply different, or novel, from those previously tried. However, in order to make such comparison, it has typically been necessary to keep a history of all previous attempts. These histories are often called *archives*. Moreover, because novelty search ignores the objective, and thus the information contained within it, it can take an a priori unknown large number of cycles to reach the objective.

Furthermore, novelty search can only be successful if the measure of novelty is appropriately defined [8]. Typically, the behaviour shown by an individual is categorised and a distance measure as an indicator for novelty distance is defined on these categories i.e., the end-point reached in a maze or the full path traveled could be defined as the behaviour, where a simple distance measure defines the novelty. However, choosing a good representation might be as difficult as choosing the right objective function.

The following section will describe the general CGA which is used here to solve the problems of novelty search discussed above.

III. Coevolutionary Genetic Algorithm

A CGA is a genetic algorithm (GA) which uses two evolving populations. This section paraphrases the description of the CGA developed by Paredis [9]:

Two populations are created. The first population is called the *solution* population, and the second population is called the *problem* population. All individuals from both populations are initialised with a uniformly random distribution. Individuals are evaluated in encounters between both the populations. Each encounter results in a behaviour. The objective function awards a score/fitness to each behaviour. However, because there are multiple solutions and problems, the overall score of an individual is determined by several encounters (in this paper 20 encounters). Each individual's total score is the aggregation of its encounters. The explicit aggregation is irrelevant as long as the total order is preserved; hence, taking either the average or the sum of scores is sufficient. Usually, a predator-prey relationship exists between the solutions and problems , although symbiotic relations are possible [9]. Therefore, the fitness score for a solution has a positive value, and the score for the problem has a negative value. Hence, the success of a solution is the failure of the problem and the other way around. Both populations are then sorted in order of their fitness value.

After the initialisation of the two populations and the calculation of their fitness, the algorithm executes the cycle described by the pseudo-code given below. First, encounters (here 20) occur between selected solutions and problems. The selection is solely influenced by an individual's rank in its population, and hence the actual fitness score is irrelevant. The advantages of rank-based selection have been shown by Whitley with his GENITOR algorithm [13]. This paper uses a selection bias of 1.5, meaning that the top ranked individuals are 1.5 times more likely to be selected than individuals in the centre of the ranking. Each encounter results in a novel fitness score which updates the total score of the involved individuals. The new fitness score removes the fitness score of the oldest encounter that is still in the history. Hence, the history of an individual's latest encounters (here 20) define its total score. The updated total score can result in a change in ranks in each population. This concludes the evaluation phase of a cycle (see first loop of the pseudo-code).

In the subsequent reproduction phase (see second loop of the pseudo-code), each population generates a single offspring, and the lowest ranked individual is removed. The same selection procedure as used in the encounter selection is applied to both populations to choose two parents each. The genetic information of both parents is combined by 2-point crossover. Furthermore, on average, a single gene mutates by shifting the gene's value by a small amount (here 0.001) with a probability (here 0.6) or is re-initialised uniformly with another probability (here 0.4) (see later in table IV). Each new individual is immediately evaluated the appropriate number of times to fill up their history by the matching process as explained above. The individual is then inserted into the sorted population according to its rank while removing the least fit individual to maintain a constant population size. In the case that the new individual is less fit than the lowest ranked individual, it is effectively not inserted in the population i.e., the population remains unchanged.

```
DO 20 TIMES
  ind1 := SELECT(pop1); ind2 := SELECT(pop2)
  payoff := EVALUATE(ind1, ind2)
  UPDATE-HISTORY-AND-FITNESS(ind1, payoff)
  UPDATE-HISTORY-AND-FITNESS(ind2, -payoff)
ENDDO

FOREACH pop IN (pop1, pop2)
  p1 := SELECT(pop); p2 := SELECT(pop)
  child := MUTATE-CROSSOVER(p1, p2)
  f := INIT-HISTORY(child)
  INSERT(child, f, pop); ENDFOREACH
```

The following section describes how combining the CGAs and novelty search can solve the problems arising in each of these approaches separately.

IV. Fitness-Diversity Driven Coevolution

Fitness-Diversity Driven Coevolution (FDDC) is the result of combining a CGA with the concept of novelty search without using an archive.

To avoid the difficulties of defining a separate novelty measure, the fitness score is reused and also defined as the novelty measure. However, while fitness is relative in the CGA context, but novelty measure generally requires an absolute measure of novelty. The relativity of the new novelty measure makes it impossible to construct an archive to compare active individuals with preceding ones from earlier cycles. However, because the problem set changes simultaneously, the archived results cannot be compared to the latest results due to the changed problem set. Hence, the novelty measure is only valid for the current state of the population and in other words simply measures the diversity of fitness values. Thus, this paper terms the new approach Fitness-Diversity Driven Coevolution (FDDC) rather than Novelty Driven Coevolution.

In order to add Fitness-Diversity to the CGA described in section III only a small number of changes are necessary.

The first change is to modify the ranking function of either the solution or the problem population. *Instead of ranking by fitness scores, the minimal difference in terms of fitness between each individual determines an individual's rank* (see equation 1 defined in [14]). The process can best be understood as a two-step process. First, the population is sorted by fitness. Then the population is re-sorted by the fitness difference towards the nearest neighbour i.e., the individual above or below in the initial sorting of the first step. Thus, the best individual is now the individual whose fitness differs the most from the fitness of the other individuals. *Hence, the optimisation process for the re-sorted population is going to maximise the differences between fitness scores rather than maximising the fitness score*. Optimisation of local fitness scores on an individual level are replaced by a population-wide Fitness-Diversity optimisation. In a single population scenario, the above describes novelty search without an archive, where the active state of the population instead maintains all *novelty* information. However, in a coevolutionary setup where the opposing population remains driven by fitness-only, the effects are different.

$$fitness_{diversity}(i) = \min_{\forall j \neq i} |fitness(i) - fitness(j)|, \quad \forall i, j \in Population \tag{1}$$

FDDC creates a strong dynamic between solution and problem population beyond the predator-prey relationship. First, in contrast to novelty search, the 'novelty' information which is solely stored in the population changes over time. Novelty is always computed relative to the current characteristics of the opposing population. The interaction between both populations defines the extent of the novelty space, and the interaction between a Fitness-Diversity driven and a fitness-driven population balances both populations in terms of fitness, thereby actively avoiding the unbalanced coevolution problem found in general CGAs.

Furthermore, FDDC actually reintroduces the overall objective which was removed in novelty search. Fitness-Diversity causes the novelty-space to expand by increasing the differences between individuals. On the other hand, the fitness driven population causes the novelty-space to move across the fitness landscape. One way to understand this is to think of pure fitness driven optimisation as water following the gradient of the 'objective function'. Once reaching a steep slope, the rest of the water (the population) is likely to follow. Fitness-Diversity, however, is better thought of as a cloud of gas, which expands in all directions. This cloud is still driven by the objective function though it moves as a whole. This way, a broader part of the search space is covered, which makes getting caught in a local extrema in the fitness landscape less likely. Hence, an algorithm using Fitness-Diversity can avoid premature convergence while still being guided by the objective function. Further, due to the coupling between the populations, this holds for both populations. The application of FDDC on sample problems in the following sections will show these effects.

V. Evolving SLIPs and terrains

The following section briefly explains the setup of the experiments by giving first the definition of the SLIPs, the terrains and the configuration of the genetic algorithm.

A. SLIPs

The SLIP model has only three morphological parameters, which are: its mass, the length of the spring and the spring constant. Furthermore, the simulation of the physics of the SLIPs (described below) requires additional variables, which are summarised in the following Table I.

Simulation Variables	
m	Mass of the spring
L	Rest length of the spring
l	Current length of the spring
k	Spring constant
α	Angle of the spring
x	Horizontal coordinate of the point mass
y	Vertical coordinate of the point mass
$\dot{x}$	Horizontal velocity of the point mass
$\dot{y}$	Vertical velocity of the point mass
g	Vertical acceleration of the gravity
sx	Horizontal coordinate of the spring tip
sy	Vertical coordinate of the spring tip
T	Terrain
$F_T(x)$	Function returning height of terrain T at x.

TABLE I
Variables of the SLIP physics simulation.

Deploying a SLIP on a terrain can be either done by dropping it from a certain height and/or initialising the model with an initial velocity vector. If a SLIP's passive or active control is well-behaved, the spring enters a following motion

cycle which can be divided into two phases, a *Flight Phase* and a *Stance Phase*, as shown in Figure 1.

Flight Phase: while the SLIP is not in contact with the ground, its motion simply follows the ballistics of its point mass. Once the tip of the SLIP touches the ground, it enters the second phase.

Stance Phase: the SLIP is now in contact with the ground. Here, the SLIP's angle α when touching the ground influences the outcome of the *Stance Phase*. This angle is usually referred to as *angle of attack* (see α in Figure 1). The point mass is still accelerating in the vertical direction due to gravity, but the spring begins to convert the kinetic energy of the SLIP system into potential energy. Once the potential energy of the spring is equal to the kinetic energy, the process gets reversed. The spring starts to expand and pushes the point mass away from the ground. The fully expanded spring converts enough kinetic energy to pull the spring off the ground and the system enters the *Flight Phase* again.

Overall the SLIP is an energy conserving system. Given the SLIP's stance point (sx,sy), the angle and current length can be computed. The state vector in (2) and the following system of differential equation (3) fully describes the motion of the SLIP during the *Stance Phase*.

$$s = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} \tag{2}$$

$$\begin{aligned} \dot{s} &= \begin{pmatrix} \dot{x} \\ \dot{y} \\ \frac{k(L-l)\sin(\alpha)}{m} \\ \frac{k(L-l)\cos(\alpha)}{m} + g \end{pmatrix} \\ &= \begin{pmatrix} \dot{x} \\ \dot{y} \\ \frac{k(L-\sqrt{hx^2+(x-F_T(sx))^2})\sin(\alpha)}{m} \\ \frac{k(L-\sqrt{hx^2+(x-F_T(sx))^2})\cos(\alpha)}{m} + g \end{pmatrix} \end{aligned}$$
$$\textbf{where } hx = x - sx \textbf{ and } \alpha = \arctan(hx, x - F_T(sx)) \tag{3}$$

For brevity the trivial ballistic equation for the *Flight Phase* is not given here, but can be found in [14].

B. Terrains

The terrain characteristic has a significant influence on the SLIPs motion and success. Therefore, a terrain generator which can produce a broad range of different terrains is required. Additionally, it is helpful to use a generator with easily controllable parameters.

The midpoint-displace algorithm used in this paper is one commonly used for 2D terrain generation in side-scrolling games [10] and is well understood. This algorithm, also known as diamond-square algorithm [15], has only four parameters: (height (h), roughness (r), displacement (D), resolution (p)). Nevertheless, it is capable of producing a broad range of terrains.

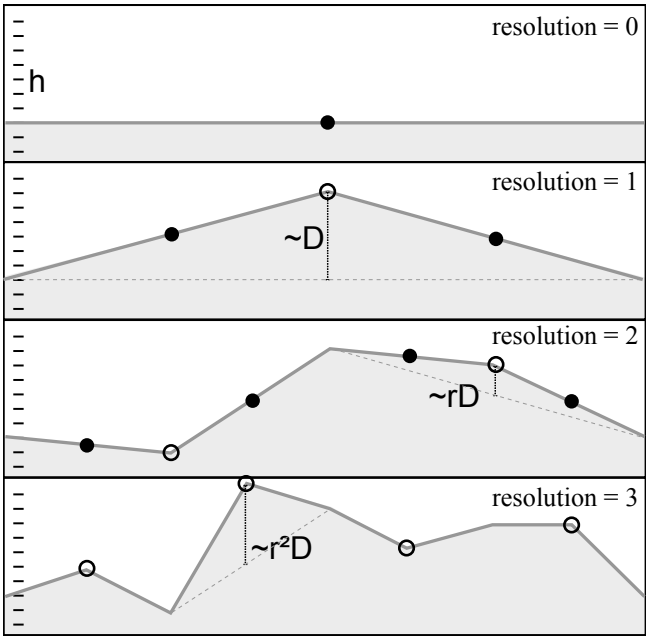

Fig. 2. Output of the terrain generator with unchanged roughness (r), base height (h) and maximal displacement (D) at different levels of resolution. Filled dots indicate midpoints used in the creation for next resolution level. Empty dots indicate the latest displaced midpoints at the current resolution level.

The implementation below is a minor adaptation of that found in [10]. The usual implementation generates terrains of finite length, while this paper applies sectioning to create terrains of infinite length. Sections are joined by taking their endpoints and forcing them to the same height, therefore, making it easier to connect adjacent sections. Next recursively all midpoints between the current points are selected and shifted according to the *displace* and *roughness* parameter. The recursion stops once a defined *resolution* is reached, Figure 2 illustrates the process of terrain generation.

C. Genetic Algorithm

The CGA evolves two populations. The first population consisting of SLIPs has three morphological characteristics: mass (m), rest length (L) and a constant spring constant term (z) of the parametrised function (5) defining the effective spring constant. The SLIP controller controls the *angle of attack* (Equation 4) taken from [16], during the *Flight Phase* and the spring constant during the *Stance Phase* (Equation 5). Both functions are parametrised and these parameters (a,b,c) are also evolved by the GA.

$$\text{angle} : \alpha = \mathbf{a}\ \dot{x} + \mathbf{b} \tag{4}$$

$$\text{spring constant} : k = \mathbf{c}\ (1 - \frac{l}{\mathbf{L}}) + \mathbf{z} \tag{5}$$

Overall the GA has six control parameters to evolve: a,b,c,m,L,z. A single, floating point value encodes each control parameter. Predefined bounds (Table II) restrict the value

range of the control parameters such that only physically and morphologically correct values are allowed.

control parameter	low bound	high bound
a $\dot{x}$ (a)	-0.5	0.5
b (b)	-0.5	0.5
c $(1 - \frac{l}{L})$ (c)	0.1	1.0
spring constant (z)	0.1	5.0
rest length (L)	0.5	3.0
mass (m)	10.0	150.0

TABLE II
BOUNDS ON THE SLIP CONTROL PARAMETERS.

The objective function, i.e. fitness, for the SLIP population used in the experiments is defined as the distance traveled in the positive direction of the x-axis. Due to the predator-prey relation between SLIPs and terrains, the fitness of the terrains is simply the negative fitness value of the encountered SLIPs.

The second population of the CGA are the terrains. All five control parameters explained in section V-B are evolved and their ranges are also bounded (Table III).

control parameter	low bound	high bound
height (h)	0.0	50.0
roughness (r)	0.0	1.0
displacement (d)	0.0	50.0
resolution (p)	0.0	10.0
seed (s)	0.0	100.0

TABLE III
BOUNDS ON THE TERRAIN CONTROL PARAMETERS.

Both GAs for evolving the SLIPs and terrains respectively use identical parameters shown in Table IV. To ensure that slightly altering the GAs parameters has no significant impact on the results obtained, a sensitivity analysis per parameter was performed.

GA Parameter	value
SLIP population size	50
Terrain population size	50
Expected number of crossovers	1
Expected number of mutations	1
Shuffle mutation probability	0.4
Change mutation probability	0.6
Delta change	0.001
History size	20

TABLE IV
PARAMETERS USED FOR THE GA.

The following section describes and discusses the results obtained.

D. Advanced CGA

The classic CGA (see section III) often leads to premature convergence. To avoid this behaviour, the advanced CGA proposed by Paredis [17] is used, where instead of fixed reproduction rates (1 per cycle), each population's reproduction rate depends on its fitness. The reproduction rate of the solution population negatively correlates with its fitness. On the other hand, the reproduction rate of the problem population positively correlates with its fitness (figure 3). The following pseudo-code replaces the second part of the pseudo-code given in section III in order to enforce the balance.

```
IF RANDOM(0.0, 1.0) < 1-
        BEST-FITNESS(pop1) / MAX-FITNESS:
    p1 := SELECT(pop1)
    p2 := SELECT(pop1)
    child := MUTATE-CROSSOVER(p1, p2)
    f := INIT-HISTORY(child)
    INSERT(child, f, pop1)
ENDIF

IF RANDOM(0.0, 1.0) <
        BEST-FITNESS(pop2) / MAX-FITNESS:
    p1 := SELECT(pop2)
    p2 := SELECT(pop2)
    child := MUTATE-CROSSOVER(p1, p2)
    f := INIT-HISTORY(child)
    INSERT(child, f, pop2)
ENDIF
```

The code above unfortunately introduces a new parameter MAX-FITNESS which in most cases is a priori unknown and must also be estimated.

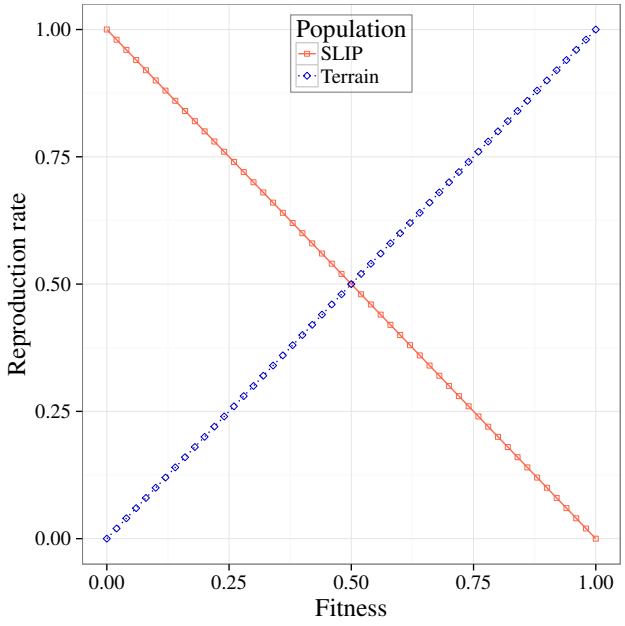

Fig. 3. Adaptive reproduction rate in the modified CGA for solution and problem population.

VI. RESULTS AND DISCUSSION

To measure the performance objectively across different runs, a fixed set of SLIPs and terrains were created by sampling their respective parameter space uniformly. These benchmark sets are then used to compute comparable performance scores. In the case of the terrains, the inverse of the

performance is used as an indicator of the terrain's difficulty. The following subsections illustrate and elaborate on the results obtained.

A. Comparing CGA and FDDC

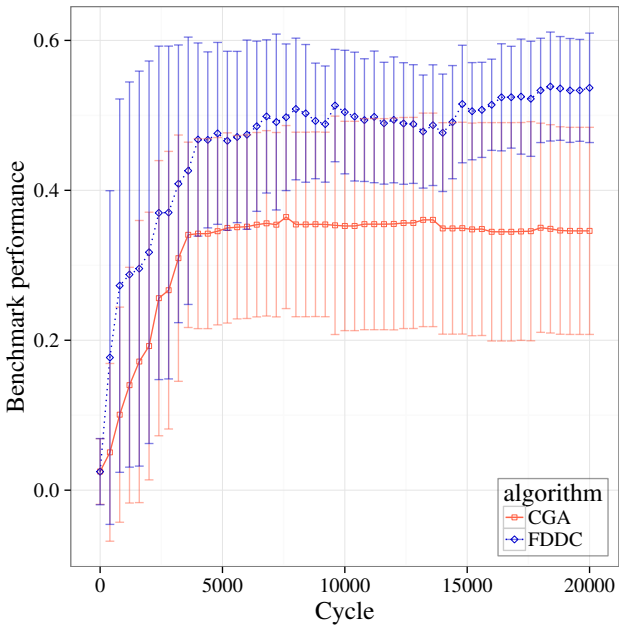

Fig. 4. Average maximum SLIP performance per cycle over 10 runs.

Figure 4 shows that Fitness-Diversity outperforms the CGA. Fitness-Diversity reaches higher benchmark scores already from the earliest cycles, and remains there throughout.

In addition, the standard deviation is lower for the FDDC than for the CGA. This indicates that FDDC produces more reliable and consistent results. Additionally, the overall performance of the FDDC algorithm converges towards a higher level than the CGA, see figure 5.

Figure 6 shows the difficulty levels of the terrain population. Here, the continuous increase of average difficulty of the CGA can be observed. The FDDC on the other hand quickly reaches a stable level of difficulty. At the same time FDDC maintains a set of very challenging and easy terrains. Moreover, the minimal, average and maximal difficulty encountered in the CGA begins to converge towards solely difficult terrains. Hence, FDDC is able to maintain a diverse set of terrain, in contrast to the CGA, which completely loses terrain diversity in the long run. Additionally FDDC is able to evolve functional SLIPs in all runs. However, the CGA was unable to evolve SLIPS in run three and six.

The density plots in figure 7 show that CGA mostly evolves populations of terrains which have very little diversity. Most of the evolved terrains are very difficult. Furthermore, in one run (run 4) the terrain difficulty did not evolve at all. This indicates that the SLIP population converged prematurely towards simple but unwanted motion, i.e. falling as far as possible rather than coordinated jumps (see the spike in run 4 in Figure 7). On the other hand, the distributions of terrains of the FDDC simultaneously contain both easy and difficult terrains.

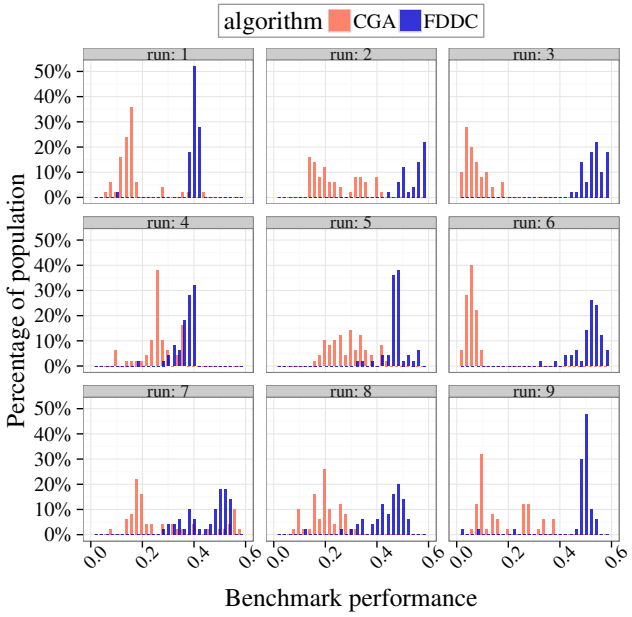

Fig. 5. Distribution of SLIP performance in the final cycle in each of 9 samples.

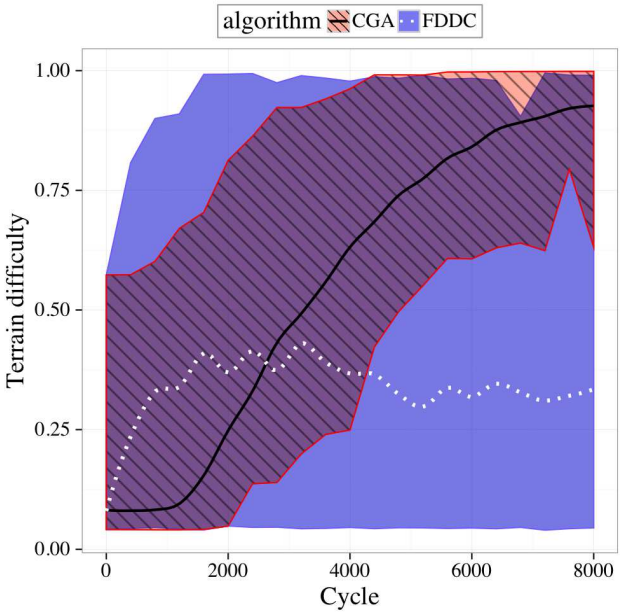

Fig. 6. Average terrain difficulty per cycle over 10 runs (solid and dotted line). The areas show the minimal and maximal average terrain difficulty for both algorithms.

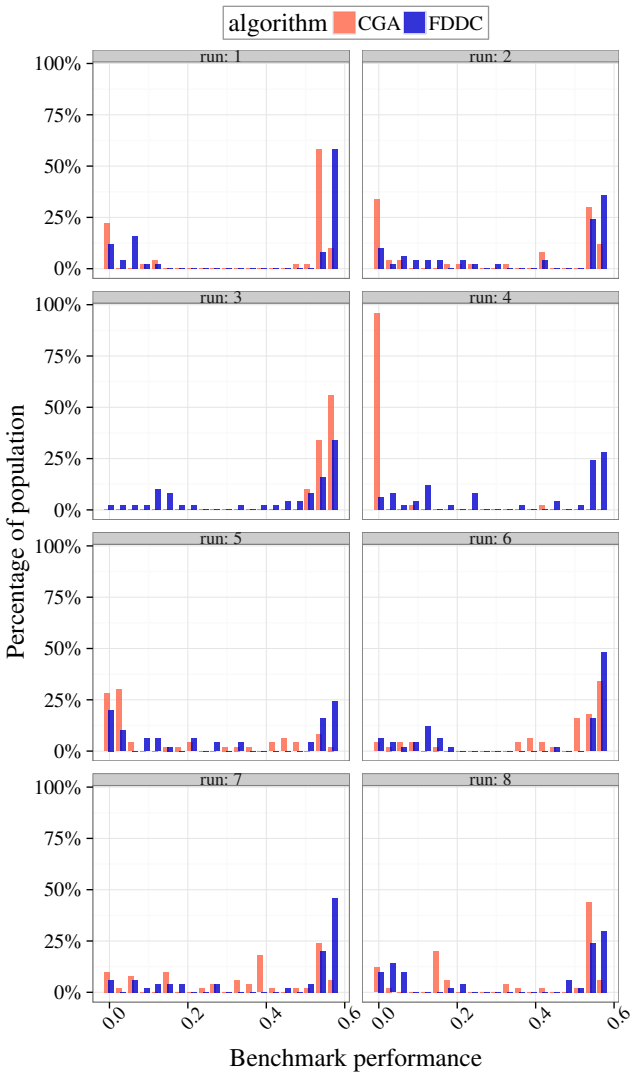

Fig. 7. Distribution of the terrain difficulty in the final cycle in each of 8 samples.

B. Differences from Random selection

Additional experiments were run where the terrain selection process was replaced by purely random selection, demonstrating that Fitness-Diversity is indeed actively creating a diverse set of terrains and that the same results cannot be achieved by a completely random process. As expected the terrains evolved by a random process were mostly trivial and did not represent a challenge for the slips, as shown in figure 8.

C. Other Novelty Measures

In the preceding experiments, novelty search was based on fitness. In the following experiments, fitness novelty was replaced by genotype novelty. The original results can be found in [14]. Here, the genotype novelty search used a Manhattan distance measure to rank individuals by normalised differences in their genotype, while keeping all other settings

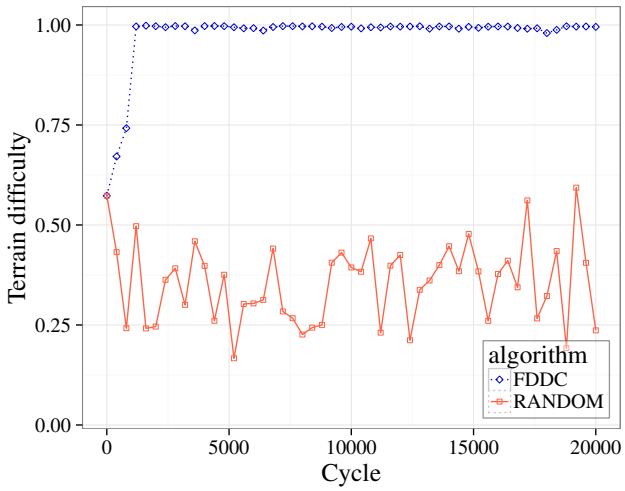

Fig. 8. Average maximal terrain difficulty per cycle over 10 runs.

identical to the other experiments. Again, an archive was not used to provide comparable conditions.

The results indicated that genotype novelty search performs significantly (p-value < 0.001) worse than Fitness-Diversity. The genotype novelty search failed due to the size of the genotype parameter space, consisting of six and five control parameters for SLIPs and terrains respectively. The Genotype novelty search was not fitness driven and therefore had only a small chance of finding the sparse sets of valid control parameters. Fitness-Diversity, remained partly fitness driven and could focus its resources on more promising areas in the parameter space. Thus, the population's sizes could be kept small for Fitness-Diversity while genotype novelty search would have required much larger population sizes.

Fitness-Diversity simply utilises existing information more efficiently than most other approaches, and requires no additional parameters or storage structures.

D. Further Experiments

This paper applies Fitness-Diversity to the problem population. However, it is also possible to apply Fitness-Diversity to the solution population, or to both the problem and solution populations simultaneously. Thus, there are two more combinations where at least one population uses Fitness-Diversity. These settings are further explored in [14], the following will briefly describe the results obtained and summarise the key findings.

Table V compares the CGA with these three different Fitness-Diversity combinations: Fitness-Diversity used on the **p**roblem population (FDDC-P i.e. the FDDC algorithm discussed in this paper), further Fitness-Diversity used for the **s**olution population (FDDC-S) and finally Fitness-Diversity applied to both **s**olution and **p**roblem population (FDDC-SP). Notice that all FDDC algorithms outperform the CGA in the long run, however the FDDC-P algorithm yields the highest performance and shows the quickest convergence.

Algorithm / Cycle	0	800	1600	2400	3200	4000
CGA (baseline)	.06	.32	.48	.50	.46	.48
FDDC-P (relative)	0.0	0.0	**+.02**	**+.08**	**+.20**	**+.22**
FDDC-S (relative)	0.0	-.08	-.06	-.08	0.0	+.02
FDDC-SP (relative)	0.0	-.20	-.08	0.0	+.04	+.14

TABLE V
BENCHMARKED SLIP PERFORMANCE OF DIFFERENT FITNESS-DIVERSITY DRIVEN ALGORITHMS RELATIVE TO THE CGA ALGORITHM. AVERAGES OVER 10 RUNS.

Further, table VI shows the average terrain difficulty for the same runs as in table V. It can be seen that the terrain difficulty of the FDDC-S setup does not converge towards very difficult terrains, even though the terrains are similarly driven by fitness like in the CGA setup. This highlights the interaction and mutual influence of both populations on each other in the Fitness-Diversity driven setups. Additionally, despite that the average terrain difficulty remains very low, the FDDC-SP setup achieves similar SLIP performance compared to the FDDC-P setup. That is, because due to the Fitness-Diversity criterion the individuals of each population are sparsely spread across the search space.

Algorithm / Cycle	0	800	1600	2400	3200	4000
CGA (absolute)	0.0	.50	.68	.56	1.0	.96
FDDC-P (absolute)	0.0	.52	.60	.52	.49	.60
FDDC-S (absolute)	0.0	.28	.60	.44	.52	.60
FDDC-SP (absolute)	0.0	0.0	0.0	.12	.12	0.0

TABLE VI
BENCHMARKED TERRAIN DIFFICULTY OF DIFFERENT FITNESS-DIVERSITY DRIVEN ALGORITHMS AND THE CGA, WHERE 0.0 MEANS EASY AND 1.0 MEANS HARD. AVERAGES OVER 10 RUNS.

These experiments show that Fitness-Diversity can improve the performance of the CGA in various ways not only by applying Fitness-Diversity to the problem population.

VII. FUTURE RESEARCH

Future research will investigate whether using Fitness-Diversity on both populations might be lead to even greater performance or flexibility improvements than already found here.

In addition, it should be possible to generalise the application of Fitness-Diversity to other domains. In this context, the author experimentally applied Fitness-Diversity in the context of robot maze navigation for evolving RNNs (recurrent neural networks) with promising initial results.

Furthermore, other diversity measures could be tested to see whether they improve the performance i.e., multiple nearest neighbours could be taken into account for determining the Fitness-Diversity scores.

VIII. CONCLUSION

This paper integrates coevolution and novelty search to create a versatile Fitness-Diversity driven algorithm called FDDC. It actively avoids the potential for deception by the objective function and is thus less likely to suffer premature convergence. Furthermore, the self-balancing dynamics exhibited by the new type of coevolution compensates for the coevolutionary problem of unbalanced coevolution. Importantly, the FDDC algorithm does not introduce extra parameters relative to the CGA and also does not require the use of an archive (as used in novelty search). Further, FDDC is applicable to CGAs in various ways. Comparison with random search shows that Fitness-Diversity significantly improves performance.

Finally, it has been shown that the FDDC algorithm is capable of evolving robust SLIP controllers while at the same time removing the need to predefine appropriate test terrains.

REFERENCES

[1] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 228–234, Jun. 1990.

[2] G. Squillero and A. Tonda, "Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization," *Information Sciences*, vol. 329, pp. 782–799, Feb. 2016.

[3] S. Ficici and J. B. Pollack, "Challenges in Coevolutionary Learning: Arms-Race Dynamics, Open-Endedness, and Mediocre Stable States," in *Proceedings of the Sixth International Conference on Artificial Life*. MIT Press, 1998, pp. 238–247.

[4] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 41–49.

[5] K. A. D. Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. dissertation, 1975.

[6] S. W. Mahfoud, "Crowding and preselection revisited," *Urbana*, vol. 51, p. 61801, 1992.

[7] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evol Comput*, vol. 5, no. 1, pp. 1–29, 1997.

[8] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evol Comput*, vol. 19, no. 2, pp. 189–223, 2011.

[9] J. Paredis, "Coevolutionary computation," *Artif. Life*, vol. 2, no. 4, pp. 355–375, 1995.

[10] J. Brown, "Simple 2d Terrain With Midpoint Displacement," http://www.somethinghitme.com/2013/11/11/simple-2d-terrain-with-midpoint-displacement/, Nov. 2013.

[11] G. Piovan and K. Byl, "Enforced symmetry of the stance phase for the Spring-Loaded Inverted Pendulum," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 1908–1914.

[12] R. Altendorfer and others, "Evidence for Spring Loaded Inverted Pendulum Running in a Hexapod Robot," in *Experimental Robotics VII*, ser. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, 2001, no. 271, pp. 291–302.

[13] D. Whitley, "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best." San Mateo: CA. Morgan Kaufmann Publishers, 1989, pp. 116–121.

[14] F. Fränz, "Fitness-Diversity driven Coevolution of Spring Loaded Inverted Pendulums and Terrains," B. Sc. Thesis, Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, unpublished manuscript, Jun. 2016.

[15] A. Fournier, D. Fussell, and L. Carpenter, "Computer Rendering of Stochastic Models," *Commun. ACM*, vol. 25, no. 6, pp. 371–384, Jun. 1982.

[16] M. Howard, "Morphology and Control Optimisation of the Spring Loaded Inverted Pendulum Model Using Genetic Algorithms," B. Sc. Thesis, Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, unpublished manuscript, Jun. 2015.

[17] J. Paredis, "Towards Balanced Coevolution," in *Parallel Problem Solving from Nature PPSN VI*, ser. Lecture Notes in Computer Science, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Springer Berlin Heidelberg, Sep. 2000, no. 1917, pp. 497–506.